VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

# Detekce SPZ pomocí obrazů

License Plate Detection Using Images

Michal Ščepka

Bakalářská práce

Vedoucí práce: Ing. Radovan Fusek, Ph.D.

Ostrava, 2021

## Abstrakt

Cílem této práce je vytvoření aplikace, která bude detekovat státní poznávací značky pomocí obrazů. Aplikace tohoto druhu může být použita například v oblasti automatických parkovacích systémů, systému pro vymáhání práva, analýzy dopravy a mnoho dalších. Dále jsou v práci popsány metody používané k detekci registračních značek. V další části jsou popsány vlastní úpravy již existujícího detektoru objektů a tvorba datasetů pro potřeby této práce. Poslední část je věnována experimentům, kde je porovnána přesnost a rychlost detekce detektorů objektů, trénovaných na různých datech a parametrech.

## Klíčová slova

počítačové vidění; detekce registračních značek; registrační značka

## Abstract

This work aims to create an application that will detect license plates using images. Applications of this kind can be used, for example, in the field of automatic parking systems, law enforcement systems, traffic analysis, and many others. Furthermore, the work describes the methods used to detect license plates. The next part describes the modifications of the existing object detector and the creation of datasets for the needs of this work. The last part is devoted to experiments where the accuracy and speed of detection of object detectors, trained on various data and parameters, are compared.

## Keywords

computer vision; license plates detection; license plate

## Poděkování

Rád bych poděkoval vedoucímu bakalářské práce Ing. Radovanu Fuskovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této práce. Děkuji také vedoucímu útvaru Strážní služba panu Pavlu Wozniakovi za konzultaci a poskytnutí kamerových záznamů z vjezdů do areálu VŠB. Dále děkuji panu Damiánu Křižkovi za poskytnutí fotografií jeho vozidla a povolení je veřejně publikovat. Na závěr bych chtěl poděkovat přátelům a rodině, zejména slečně Monice Kempové, za velkou podporu.

# Obsah

# Seznam použitých zkratek a symbolů

| | | |
|---|---|---|
| AP | – | Average Precision |
| ARRZ | – | Automatické rozpoznávání registračních značek |
| CNN | – | Convolutional neural network |
| COCO | – | Common Objects in Context |
| FPS | – | Frames per second |
| GC | – | Google Colaboratory |
| GPU | – | Graphics Processing Unit |
| IoU | – | Intersection over Union |
| mAP | – | Mean Average Precision |
| NMS | – | Non-maximum Suppression |
| OI | – | Open Images |
| Pascal VOC | – | Pascal Visual Objects Classes |
| px | – | pixel |
| R-CNN | – | Region Based Convolutional Neural Network |
| RPN | – | Region Proposal Network |
| RZ | – | Registrační značka |
| SPZ | – | Státní poznávací značka |
| YOLO | – | You Only Look Once |

# Seznam obrázků

# Seznam tabulek

# Kapitola 1

# Úvod

Automatické rozpoznávání registračních značek (ARRZ) má širokou škálu aplikací, protože registrační značka (RZ) je primární, nejrozšířenější, člověkem čitelný, povinný identifikátor motorových vozidel. Rozpoznávání RZ poskytuje automatizovaný přístup k obsahu poznávací značky pro počítačové systémy spravující databáze a zpracovávající informace o pohybech vozidel. Tato práce je členěna na následující kapitoly: úvod do problematiky, metody detekce RZ, vlastní řešení a experimenty. U obrázků 2.3, 3.1 a 4.4 majitel vozidla svolil možnost publikování RZ. Níže jsou uvedeny některé z hlavních aplikací ARRZ:

## Parkování

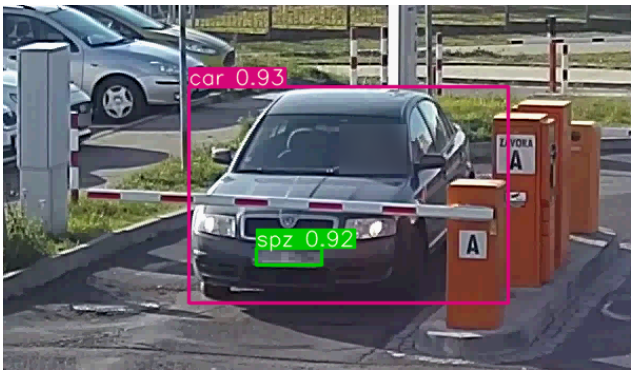
Jednou z hlavních aplikací ARRZ je automatizace parkování a zabezpečení parkování: správa parkovacích poplatků bez lístků, automatizace přístupu k parkování, navádění k umístění vozidla, prevence krádeží automobilů, podvody se „ztracenými lístky“, podvody při výměně lístků, automatizovaný platební proces a mnoho dalších. [1]

Například nově od roku 2021 je vjezd na parkoviště Vysoké školy báňské (VŠB) pro každého držitele čipové karty, který bude chtít využít parkoviště v areálu, povinnost si do systému zaevidovat RZ vozidla, kterým chce vjíždět na zmíněné parkoviště. Jinak nebude řidič s vozidlem do areálu vpuštěn.

Na obrázku 1.1 je zobrazen snímek závory u vjezdu do areálu parkoviště VŠB, zpracovaný detektorem RZ vytvořeným v této práci.

## Řízení přístupu

Řízení přístupu je obecně mechanismus omezující přístup k oblastem a zdrojům na základě identit uživatelů a jejich členství v různých předdefinovaných skupinách. Přístup do omezených zón však




Obrázek 1.1: Kamerový snímek závory u vjezdu do areálu parkoviště VŠB na němž je konvoluční neuronovou sítí rozpoznáno vozidlo a jeho RZ, která byla následně rozmazána.

může být také řízen na základě vozidel samostatně nebo společně s identitou člověka. ARRZ přináší automatizaci řízení přístupu vozidel na parkovištích nebo v obytných oblastech.

## Mýtné na dálnicích

Silniční mýtné motoristé platí za používání konkrétního segmentu silniční infrastruktury. Mýtné je běžným způsobem financování výstavby silnic a dálnic. Efektivní výběr mýtného omezuje podvody spojené s jeho neplacením. ARRZ se většinou používá jako velmi účinný nástroj prosazování zda řidič motorového vozidla zaplatil mýtné.

## Hraniční kontroly

Hraniční kontrola je státem koordinované úsilí o dosažení kontroly nad státními hranicemi země s prioritním posláním podporovat bezpečnost státu. ARRZ přidává významnou hodnotu zaznamenáváním událostí, vytvářením databází o vozidlech překračujících hraniční přechody.

## Analýza dopravy

Měření doby jízdy je velmi efektivní metoda pro porozumění provozu, detekci nápadných situací, událostí atd. Například čas cesty vozidla lze spolehlivě měřit systémy založenými na ARRZ. Data shromážděná těmito systémy lze po zpracování použít pro informování řidičů o dopravní situaci ke zvýšení bezpečnosti silničního provozu.

## Vymáhání práva

ARRZ je ideální technologií pro účely vymáhání práva. Je schopno automaticky identifikovat odcizené vozy na základě aktuální veřejně dostupné databázi registru odcizených vozidel. Pokud v ní je vozidlo se stejnou RZ, je daná skutečnost archivována a je upozorněn operátor. Databáze je denně automaticky aktualizována. Systém ukládá záznamy o všech vozidlech projíždějících pod kamerou, včetně záznamů o vozidlech, o nichž v době čtení není známo, že by byly kradené. Aby bylo možno je zpětně vyhledat pro případné vyšetřovací účely. Systém rovněž odhaluje neoprávněné používání registračních značek. Dalšími velmi běžnými aplikacemi pro vymáhání práva je kontrola jízdy na červenou a překročení maximální povolené rychlosti. [2]

V Londýně ve Velké Británii se rovněž dá pomocí ARRZ vymáhat pokuta, pokud vozidlo, které k tomu nemělo povolení, použilo pruh určený pro autobusy. [3]

## Změny v RZ

V roce 2002, se kvůli systémům ARRZ změnil styl nizozemských registračních značek. Jedna ze změn se dotkla stylu písma, kde se přidaly malé mezery do písmen P a R, aby byly lépe rozlišitelné těmito systémy. [4]


Obrázek 1.2: Změněný styl písma na nizozemských RZ.

# Kapitola 2

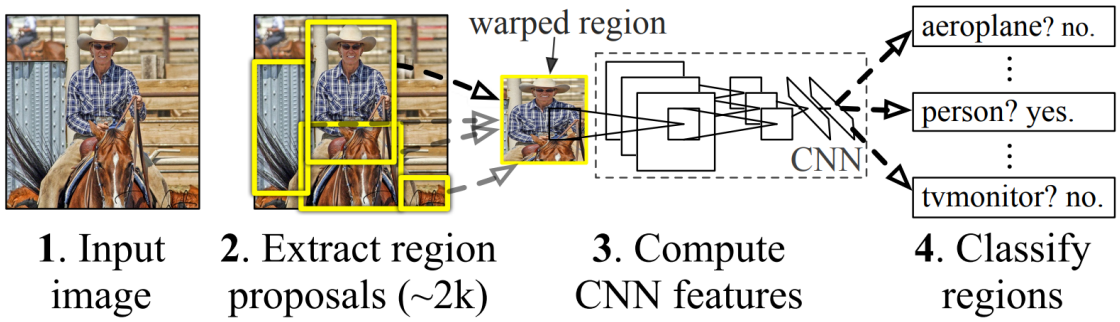
# Metody pro detekci RZ

V této kapitole budou popsány některé techniky detekcí RZ. V posledních letech se k detekci RZ používají převážně konvoluční neuronové sítě. Detekce objektů je technologie počítačového vidění, která se zabývá nalezením a klasifikováním objektů ve snímcích.

V následujících podkapitolách budou popsány detektory objektů YOLO (You Only Look Once) a R-CNN (Region Based Convolutional Neural Network). Byly vybrány tyto dva detektory, protože k oběma již byla vytvořena spousta publikací na téma detekce RZ a také vlastní řešení této práce se bude věnovat YOLO detektoru.

## 2.1 Region Based Convolutional Neural Network (R-CNN)

R-CNN [5] detektor objektů se skládá ze tří modulů. První pomocí selektivního prohledávání (selective search) generuje návrhy regionů (region proposals) nezávisle na kategorii objektů. Tyto návrhy definují sadu oblastí, ve kterých detektor bude dále hledat objekty. Těchto oblastí je okolo 2000. Druhým modulem je konvoluční neuronová síť (CNN), která zpracuje každou oblast. Třetí modul vezme výstup z CNN a klasifikuje tyto regiony. Tento postup je popsán na obrázku 2.1.




Obrázek 2.1: Detektor objektů R-CNN (zdroj [5]).

### 2.1.1 Fast R-CNN

Fast R-CNN [6] zvyšuje rychlost detekce. Již se neskládá ze 3 modulů, ale pouze z jednoho. Provádí extrakci vlastností (feature extraction) ještě před vygenerováním návrhů regionů, takže CNN projde celý obrázek pouze jednou, namísto zpracovávání 2000 regionů v obrázku. Největší časovou náročnost v detekci má použitý algoritmus selektivního vyhledávání.

### 2.1.2 Faster R-CNN

Faster R-CNN [7] je modifikovaná verze Fast R-CNN. Hlavním rozdílem je, že Faster R-CNN namísto algoritmu selektivního prohledávání využívá rychlou neuronovou síť RPN (Region Proposal Network) pro vygenerování návrhů regionů.

## 2.2 You Only Look Once (YOLO)

Princip YOLO [8] modelu spočívá v tom, že narozdíl od ostatních detektorů založených na technice posuvného okénka (sliding window) nebo návrhu regionů (region proposals), model YOLO nemusí redundantně procházet obrázek, aby byl schopen najít všechny objekty. YOLO již při trénování vidí celý snímek a učí se kontext celého obrázku. Konvoluční neuronová síť se podívá na obrázek pouze jednou a predikuje objekty v něm obsažené. Jedná se tedy o jednofázový detektor. To je důvod proč autoři této architektury zvolili název You Only Look Once.

### 2.2.1 Postup detekce YOLO algoritmu

Nejdříve YOLO algoritmus rozdělí obrázek na mřížku o velikosti $S \times S$. Například na obrázku 2.2 má $S$ hodnotu 7. Buňky v mřížce napomáhají detekovat objekty. Každá buňka, ve které se nachází střed objektu, je zodpovědná za jeho detekování.

Každá buňka predikuje $B$ regionů (bounding boxes) a jejich skóre (confidence score). Skóre odráží, jak moc si je model jistý, že daný region obsahuje objekt. Pokud v predikovaném regionu není objekt, jeho skóre by se mělo rovnat nule. Region může být větší, než buňka pro kterou byl vytvořen, takže může zasahovat do okolních buněk, nicméně střed pořád musí mít v původní buňce.

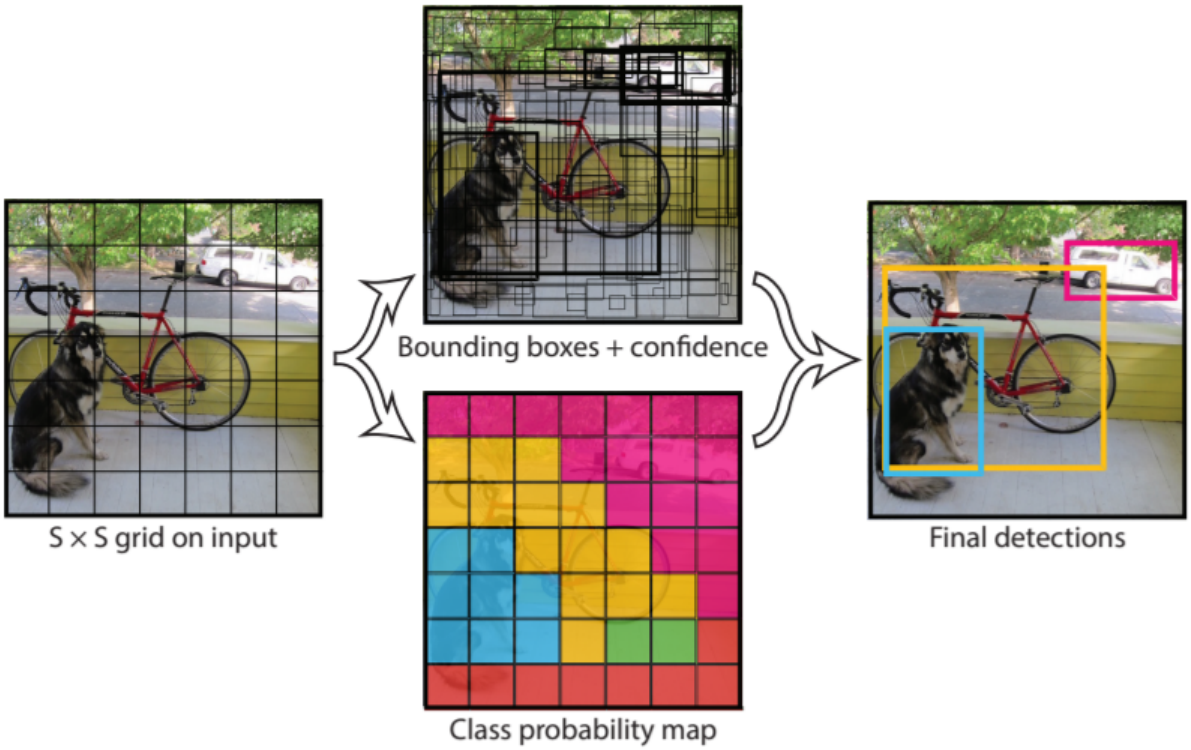
Každý region obsahuje těchto 5 hodnot: $x$, $y$, šířka, výška a skóre. Souřadnice $x$ a $y$ reprezentují střed regionu relativní k buňce, pro kterou byl predikovaný. Šířka a výška regionu jsou relativní k velikosti celého obrázku. Do finální predikce se vyberou jen regiony s konfidencí větší než je definovaný práh.

Každá buňka také předpovídá $C$ pravděpodobnost, ke které třídě regiony náleží (class probability).

Predikce jsou reprezentovány jako tensor $S \times S \times (B \cdot 5 + C)$. Například pro vyhodnocení YOLO modelu na Pascal VOC datasetu byly použity hodnoty: $S = 7, B = 2$. Pascal VOC má 20 tříd

objektů takže $C = 20$. Po dosazení do vzorce mají finální predikce na tomto datasetu tensor o velikosti $7 \times 7 \times 30$.

Více informací o postupu detekce YOLO algoritmu je možné získat z jeho dokumentace [8].




Obrázek 2.2: Detektor objektů YOLO (zdroj [8]).

### 2.2.2 Non-maximum Suppression (NMS)

Stává se, že model detekuje pro stejný objekt více regionů, ale uživatel chce na výstupu pouze jeden. NMS [9] algoritmus řeší tento problém odebráním redundantně predikovaných regionů pro stejný objekt.

Algoritmus vybere region s nejvyšší konfidencí z množiny regionů, které se překrývají a detekovaly objekt stejné třídy. Postupně vypočítá IoU mezi vybraným regionem a všemi ostatními, které se s ním překrývají. Odstraní regiony, se kterými má ten zvolený větší IoU než je stanovená hranice (obvykle $IoU \geq 0,3$).


Obrázek 2.3: Vlevo je snímek detekce před aplikací NMS algoritmu a vpravo po aplikaci.

### 2.2.3 Kotevní oblasti (Anchor Boxes)

Kotevní oblasti jsou předdefinovaná množina obdélníkových regionů s určitými rozměry. Tato množina zachycuje velikost a poměr stran obdélníkových oblastí specifické třídy, typicky se vytváří z trénovacích dat při trénování modelu. Využívá se, když je více objektů různých tříd nalezeno v jedné buňce, aby se podle tvaru kotevní oblasti rozhodlo, k jaké třídě detekovaný objekt náleží. [10]

### 2.2.4 YOLOv1

Podobně jako R-CNN detektory i YOLO prošlo určitým vývojem od svého vzniku v roce 2015. Následujících několik podkapitol shrne nejdůležitější změny v YOLO verzích.

První verze YOLO sítě má celkem 26 vrstev: 24 konvolučních následovanými dvěma plně spojenými (Fully Connected). YOLOv1 provádělo mnoho lokalizačních chyb, zvláště u malých objektů ve skupinách jako je například hejno ptáků.

### 2.2.5 YOLOv2/YOLO9000

YOLOv1 provádí mnoho lokalizačních chyb a má relativně nízký recall. Tudíž se autoři ve druhé verzi [11] zaměřili na zlepšení těchto nedostatků při snaze zachovat přesnost klasifikace. Pro dosažení lepších výsledků použili následující úpravy:

- Počet vrstev neuronové sítě se zvýšil na 30.
- Byly přidány kotevní oblasti (Anchor Boxes).
- Neuronová síť již neobsahuje plně spojené vrstvy.

### 2.2.6 YOLOv3

Jedna ze změn v YOLOv3 se týkala páteřní sítě (backbone). Namísto Darknet-19 začalo YOLOv3 používat Darknet-53, tato síť je hlubší a efektivnější. Zlepšilo se mAP detekce malých objektů. Více informací může být nalezeno v publikaci [12].

### 2.2.7 YOLOv4 a YOLOv5

YOLOv4 opět zvyšuje AP a rychlost detekce oproti YOLOv3. AP je zvýšeno o 10 % a snímková frekvence o 12 %. Více informací může být nalezeno v publikaci [13].

Pátá verze YOLO byla vydána jen pár měsíců po vydání YOLOv4. Oproti ostatním verzím je YOLOv5 implementováno ve frameworku PyTorch.

## 2.3 Detekce RZ v jiných publikacích

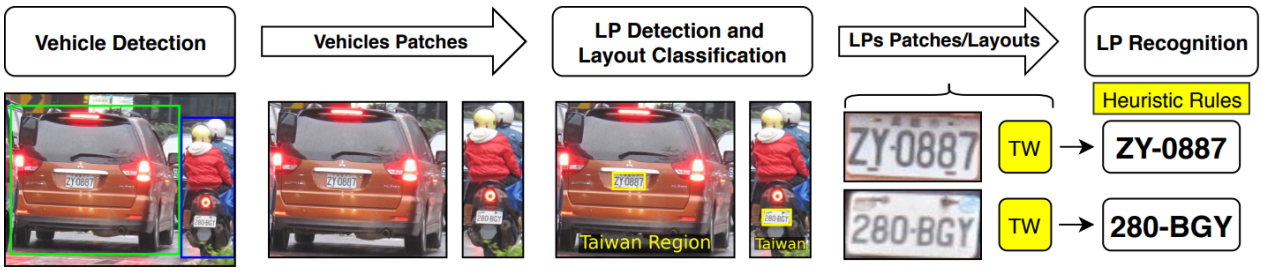
Jak již bylo zmíněno na začátku této kapitoly, existuje mnoho článků zabývajících se detekcí RZ. Liší se různými přístupy a použitými technologiemi. Některé z nich teď budou krátce popsány.

### 2.3.1 Třífázový detektor

Publikace univerzity v brazilském státě Minas Gerais z roku 2018 [14] a z roku 2021 [15] popisují jejich systémy pro detekci RZ v reálném čase postavené na upravených YOLO architekturách. Autoři publikací zvolili přístup, kdy nejdříve jeden model detekuje vozidlo, druhý v nalezeném vozidle detekuje RZ a třetí v RZ provede segmentaci a rozpoznání znaků (obrázek 2.4).

Pro detekci vozidel mají nastavenou konfidenci detekce na 0,25 a více. Pro detekci RZ zvolili nejnižší konfidenci 0, kvůli případům, kdy je RZ detekována, ale s nízkou konfidencí. Pokud je v jednom vozidle detekováno více objektů jako RZ, ponechají pouze detekci s nejvyšší konfidencí, protože vozidla mají zpravidla pouze jednu RZ.

Jejich systém na UFPR-ALPR datasetu [16] dosáhl lepších výsledků než některé komerční detektory v té době. Komerční systémy Sighthound a OpenALPR nepřesáhly úspěšnost přes 70 %, zatímco systém publikace [14] dosáhl 78% úspěšnosti s rychlostí detekce 35 snímků za sekundu.




Obrázek 2.4: Postup detekce a rozpoznání znaků RZ v publikaci [14].

### 2.3.2 Detektor bhútánských RZ

V publikaci [17] zvolili autoři odlišný přístup. Pro detekci vozidla a RZ používají jeden YOLO model a systém v nalezené RZ dále nerozpoznává její znaky. Autoři publikace si dali za cíl vyřešit problém detekce bhútánských RZ. Systémy vyvinuté pro jiné země na jejich RZ nefungují, protože jich mají

celkem 8 druhů různých tvarů a barev (obrázek 2.5). Na svém datasetu dosáhli až 98% úspěšnosti detekce.


Obrázek 2.5: Vzorek bhútánských RZ z publikace [17].

### 2.3.3 Detekroty taiwanských RZ

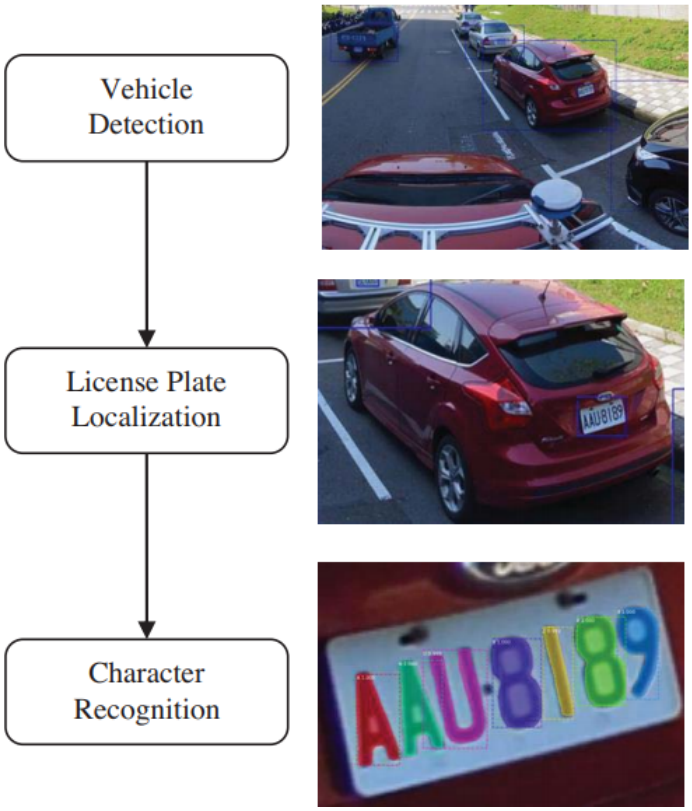
V publikaci [18] se její autoři zabývají detekcí taiwanských RZ. Tento problém řešili kombinací detekce RZ YOLO modelem a následně detekce znaků pomocí techniky posuvného okénka. Jejich systém pro detekci a rozpoznání znaků jedné RZ potřebuje 800 ms až 1 s.

Autoři v publikaci [19] jedné taiwanské univerzity popisují jejich systém na detekci RZ. Tento sytém detekuje pouze RZ, nezabývá se detekcí znaků. Autoři článku použili a upravili YOLOv1 a YOLOv2 modely. Například upravili rozměr mřížky rozdělující zpracovávaný snímek. Namísto standardního rozměru $7 \times 7$, jejich mřížka má velikost $11 \times 11$. Síť dále upravili tak aby každá buňka v mřížce mohla detekovat pouze jeden region. Zatímco originální YOLO model dosáhl na jejich testovacích datech snímkové frekvence 45 FPS, upravená verze dosáhla 54 FPS.

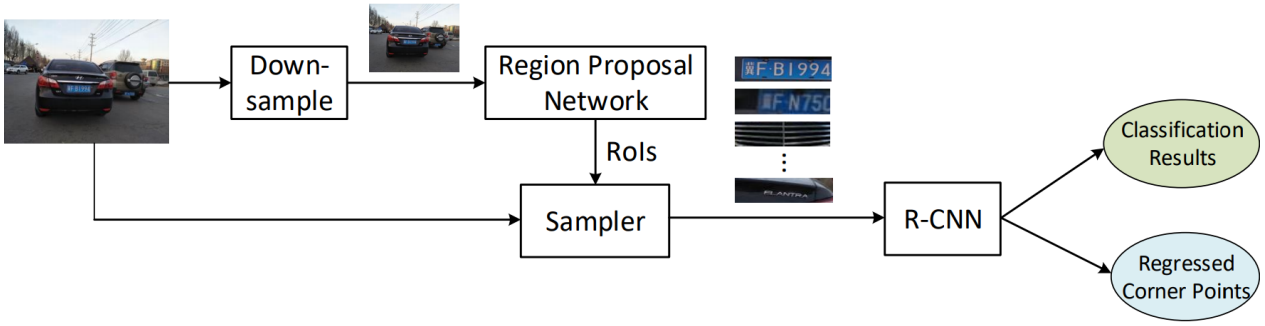
Další taiwanský článek z roku 2019 [20] zabývající se detekcí RZ, tentokrát zvolil třífázový přístup kombinující modely YOLO a R-CNN (obrázek 2.6). Jejich algoritmus v první fázi pomocí YOLOv2 sítě na snímku rozděleném do mřížky $19 \times 19$ detekuje vozidlo. Ve druhé fázi se ve vozidle pomocí další YOLOv2 sítě detekuje RZ. Nakonec ve finální fázi je použita síť Mask R-CNN pro rozpoznání znaků. Pokud je RZ na snímku dostatečně velká (zabírá přes 20 % snímku), systém může přeskočit hned na 3. fázi a detekovat znaky. Systém dosáhl v jejich testech až 91% úspěšnost.

### 2.3.4 Upravené R-CNN

Článek čínské univerzity z roku 2017 [21] popisuje detekci RZ na modelu R-CNN s upraveným algoritmem návrhu regionů (region proposal). Algoritmus nejdříve vstupní snímek převzorkuje/-komprimuje (downsample) a vygeneruje z něj regiony kandidátů na detekci RZ. Poté vzorkovač (sampler) extrahuje tyto regiony z originálního snímku ve vysokém rozlišení a pošle je do R-CNN sítě k dalšímu zpracování (obrázek 2.7). Detektor dosáhl v jejich testech $AP = 0.781$.




Obrázek 2.6: Postup detekce RZ z publikace [20].




Obrázek 2.7: Postup detekce RZ z publikace [21].

# Kapitola 3

# Metriky pro hodnocení modelů

Pro zjištění, který model dosahuje lepších výsledků na stejných testovacích datech, bez nutnosti procházet ručně každý obrázek zvlášť a subjektivně opticky porovnávat, se používají metriky Intersection over Union (IoU), Precision (Přesnost), Recall a Mean Average Precision (mAP). V pozdějších kapitolách se budou vyhodnocovat modely právě podle těchto metrik. Proto v této kapitole budou tyto čtyři pojmy vysvětleny:

## 3.1 Intersection over Union (IoU)

Jak je uvedeno v článku [22] Intersection over Union je metrika hodnotící přesnost detektorů objektů. Měří, jak moc modelem predikovaná oblast překrývá člověkem označenou oblast na objektu v obrázku. Jakýkoliv algoritmus, jehož výstupem jsou oblasti označující nalezené objekty může být hodnocený podle této metriky.

Pro vyhodnocení IoU jsou zapotřebí tyto dvě věci:

1. Člověkem označená oblast, kterou chce, aby detektor objektů při vyhodnocování obrázku označil také.

2. Detektorem objektů predikovaná oblast na obrázku.

Spočítá se průnik a sjednocení mezi člověkem označenou oblastí ($A$) a oblastí označenou algoritmem ($B$). Nakonec se IoU vypočítá vydělením oblasti průniku s oblastí překrytí:

$$IoU = \frac{|A \cap B|}{|A \cup B|}$$

Pro výsledné $IoU$ platí $0 >= IoU <= 1$. Pokud $IoU = 1$, tzn. že obě oblasti leží na totožných pozicích. Naopak při $IoU = 0$ se oblasti žádnou částí nepřekrývají. Proto je vyžadováno, aby se $IoU$ co nejvíce blížilo k 1.

Jako pravdivě pozitivní detekce se v mnoha případech považuje již $IoU \geq 0,5$. Na druhou stranu, když $IoU < 0,5$ je tato detekce považována za falešně pozitivní. Pokud je v obrázku objekt, nicméně model ho nedetekuje, jedná se o falešně negativní detekci. Pravdivě negativní případy se v detekci objektů neřeší.

Na obrázku 3.1 je příklad predikovaného rámečku a člověkem označený rámeček.


Obrázek 3.1: Příklad regionů: zelený – predikovaný modelem s konfidencí 0.87, červený – označený člověkem s vypočítaným IoU 0.77.

## 3.2 Precision (Přesnost)

Přesnost [23] dává vědět, pokud model něco detekuje, jak často je tato detekce pravdivě pozitivní. Přesnost ($P$) modelu je definována jako poměr mezi pravdivě pozitivně detekovanými objekty a celkovým počtem detekovaných objektů. Do celkového počtu detekovaných objektů patří jak pravdivě pozitivní ($PP$) tak falešně pozitivní ($FP$) detekce. Vzorec je následující:

$$P = \frac{PP}{PP + FP}$$

Přesnost modelu se snižuje, pokud model provádí hodně falešně pozitivních detekcí nebo málo pravdivě pozitivních detekcí. Naopak se přesnost zvyšuje, pokud model provádí hodně pravdivě pozitivních detekcí nebo málo falešně pozitivních detekcí.

Například pokud je $n$ obrázků, na kterých je $m$ státních poznávacích značek a po zpracování obrázků modelem vyjde výsledek, že model našel 100 RZ. Z toho 90 je pravdivě pozitivních ($PP = 90$) a 10 falešně pozitivních ($FP = 10$). Přesnost tohoto modelu se vypočítá dosazením do vzorce: $P = \frac{90}{90+10} = 0,9$. Zjistilo se, že model má přesnost $P = 0,9$.

## 3.3 Recall

Jenom přesnost jako metrika nestačí, protože neuvažuje celkový počet objektů v testovacích datech. Takže pokud bude například testovací množina s 900 RZ v $n$ obrázcích a model najde 100 RZ z

toho 90 pravdivě pozitivních. Tak přesnost bude stále $P = 0,9$ a nebude zjištěno, že model našel sotva 10 % pravdivě pozitivních objektů. Tento problém řeší recall.

Recall [23] říká, jestli model našel objekt pokaždé když ho měl najít. Čím větší je recall, tím více objektů z testovací množiny model pravdivě pozitivně detekoval. Recall ($R$) je poměr mezi pravdivě pozitivně detekovanými objekty a celkovým počtem objektů v testovacích datech. Celkový počet objektů se rovná součtu pravdivě pozitivních ($PP$) a falešně negativních ($FN$) detekcí. Vzorec je následující:

$$R = \frac{PP}{PP + FN}$$

Například když budou existovat testovací data se 150 RZ, model v nich najde 90 pravdivě pozitivních a falešně negativních $FN = 150 - 90 = 60$. Výsledný recall bude $R = \frac{90}{90+60} = 0,6$.

## 3.4 Mean Average Precision (mAP)

Precision a recall jsou rozdílné metriky a obě jsou důležité. Čím více model najde objektů, tím více ztrácí precision, ale na druhou stranu se mu zvyšuje recall. Z těchto metrik se vytváří *Precision-Recall Curve* na které se tohle chování dá pozorovat a ze které se dále vypočítává *Average Precision (AP)* [23]. AP je vypočítaný obsah oblasti pod touto křivkou. AP se počítá zvlášť pro každou třídu, kterou se model učí rozpoznávat. mAP se získá vypočítáním AP pro každou třídu a výsledky se zprůměrují. mAP se počítá pro různé hranice IoU. Například:

- mAP@0.5 znamená, že mAP bylo vypočítáno pro hranici $IoU \geq 0.5$.
- mAP@0.5:0.05:0.95 znamená, že je vypočítáno pro hranice $IoU \geq 0.5$ až po $IoU \geq 0.95$ se skokem $IoU = 0.05$. Takže pro IoU 0.5, 0.55, 0.6,..., 0.95 a všechny tyto vypočítané mAP@0.5 až mAP@0.95 byly zprůměrovány.

mAP@0.5:0.05:0.95 se dá vyjádřit následujícím vzorcem, kdy $N$ je počet iterací, v tomto případě $N = 10$.

$$mAP = \frac{\sum_{i=1}^{N} AP_i}{N}$$

# Kapitola 4

# Vlastní řešení

Na základě nastudovaných metod v předchozí kapitole jsem se rozhodl v této práci použít model YOLO, který se mi zdá svou rychlostí a přesností detekce vhodný pro tento typ úkolu. Porovnal jsem verzi YOLOv3 s YOLOv5. Zvolil jsem pátou verzi YOLO, protože době psaní této práce byla nejnovější. Dále jsem pro porovnání zvolil YOLOv3, protože je zatím nejrozšířenější, nejlépe zdokumentovaná a mezi čtvrtou a pátou generací je časový rozdíl vydání pouze pár měsíců.

Po nastudování informací o YOLO modelech jsem vyhledal jejich implementace, ze kterých jsem i vycházel při následném rozšiřování funkčností. Nejvíce mi vyhovovaly implementace v knihovně PyTorch, protože narozdíl od implementací v TensorFlow se mi podařilo tyto modely zprovoznit bez větších potíží. Modely implementované v knihovně TensorFlow mají například problémy se zpětnou kompatibilitou verzí TensorFlow 1.x a 2.x. Naneštěstí ani nainstalování správné verze nezaručuje, že model bude na počítači fungovat.

Proto pro mě bylo příjemným zjištění, že verze YOLOv5 je realizovaná právě v PyTorch. Podařilo se mi vyhledat i verzi YOLOv3 postavenou na PyTorch knihovně. Implementace YOLOv3 a YOLOv5 jsem stáhl z github repozitářů: [24] [25].

## 4.1 Rozšíření YOLO implementací

RZ je svým tvarem a obsahem podobná i objektům, které se na vozidlech nenachází. Jako je například svislé dopravní značení a plno dalších objektů. Proto pro experiment byly stažené YOLO implementace upraveny na dvoufázové řešení (podobně jako v publikaci [17]), tak aby algoritmus v první fázi detekoval vozidla a následně v druhé fázi se v nalezených vozidlech pokusil detekovat jejich registrační značky.

Tohle řešení eliminovalo možnost detekovat RZ mimo vozidlo, pokud bylo vozidlo v první fázi detekováno správně. Na druhou stranu, pokud algoritmus v první fázi nedetekoval vozidlo, tak následně nemohl přejít do druhé fáze a v ní zkoušet detekovat RZ. Bohužel i v samotných vozidlech se nachází objekty podobné poznávacím značkám, těmi jsou například světlomety. Proto když síť

detekuje ve vozidle více objektů, tak algoritmus jako výslednou RZ zobrazí tu, která měla nejvyšší skóre.

YOLO implementace byly dále rozšířeny o tyto následující funkčnosti:

- Byla přidána možnost rozmazání nalezené RZ. Tato funkce se dá vyžít například k veřejnému publikování výsledků detekce, kdy se zachová anonymita vozidel.
- Pro vyhodnocení úspěšnosti detekce byla naimplementována funkce pro počítání metrik IoU, precision, recall a AP na testovací množině. Tato funkce významně pomohla při vyhodnocování experimentů.
- Protože při detekci objektů se dá porovnávat i její rychlost. Byla přidána funkce pro zobrazení odhadovaného počtu snímků za sekundu real-time detekce videa.

## 4.2 Datasety

Datasety pro detektory objektů jsou kolekce snímků s označenými regiony, kde se podle člověka nachází požadované objekty, které se detektor učí rozpoznávat. Podobně jako člověk od narození trénuje svůj mozek rozpoznávat objekty na množství vzorků, které vidí svýma očima, se musí trénovat i detektor objektů. [26]

Obrázky v datasetech jsou rozděleny do tří kategorií:

1. První kategorie je největší, jedná se o množinu obrázků určenou pro trénovaní, obsahuje okolo 70 % z celkového počtu obrázků v datasetu.
2. Druhá kategorie je validační. Tato množina se využívá při trénovaní pro vyhodnocení jak se daná síť po každé trénovací epoše mění, vypočítáním metriky mAP a porovnáním s předchozí nejlepší epochou. Obsahuje okolo 20 % obrázků z původního datasetu.
3. Poslední množina na testování je nejmenší, obsahuje zbytek obrázků z datasetu, což je okolo 10 %. Používá se pro vlastní testování natrénovaného modelu.

Pro tuto práci byly vytvořeny celkem čtyři datasety, dva s označenými registračními značkami a dva s označenými vozidly. Vždy byl vytvořen jeden základní dataset a k němu i augmentovaná verze. Datová augmentace bude popsána v pozdější podkapitole. Datasety s vozidly byly vytvořeny z důvodu rozšíření implementace o dvoufázovou detekci.

Použité snímky v datasetech se snaží nejlépe odrážet situace, ve kterých budou výsledné natrénované sítě využívány. Situace jako například fotografie zaparkovaných vozidel, videa z palubní kamery vozidla nebo záběry z kamer do vjezdů parkovišť.

### 4.2.1 Základní dataset RZ

Pro vytvoření datasetu s registračními značkami byly využity tři datasety, které jsou volně ke stažení:

1. První dataset [27] obsahoval okolo 400 snímků automobilů, které již měly označené regiony, kde se RZ nachází. Stačilo jen převést soubory s anotacemi z Pascal VOC XML formátu do YOLO TXT formátu.

2. Druhý dataset [28] byl z jiného studentského projektu a obsahoval cca 500 fotografií zadních částí automobilů a nákladních vozidel (obrázek 4.1), nicméně k nim nebyly dostupné anotace. Takže pro tuto práci byly anotace doplněny ručně pomocí open source nástroje labelImg [29].

3. Jako třetí byl použit dataset automobilů ze Stanfordovy univerzity [30], který obsahoval 16 000 snímků (obrázek 4.2). Bylo z něj vybráno okolo 700 snímků a také ke každému ručně doplněno označení regionu s RZ.

Obrázek 4.1: Ukázka snímků z datasetu [28].

Obrázek 4.2: Ukázka snímků z datasetu ze Stanfordovy univerzity [30].

U snímků stažených s anotacemi bylo zkontrolováno, jak jsou RZ označeny. Cílem bylo, aby v každém označeném regionu byla obsažena celá RZ a zároveň aby měly všechny regiony co nejmenší nutnou velikost.

Z těchto tří datasetů ([27], [28] a [30]) byl vytvořen pro tuto práci základní dataset RZ. Základní dataset obsahuje celkem 1 590 snímků. Z toho 1 113 v trénovací, 318 ve validační a nakonec 159 v testovací množině.

### 4.2.2 Základní dataset vozidel

Pro realizaci úpravy algoritmu detekce na dvoufázovou, kdy jedna YOLO síť má za úkol detekovat vozidlo a druhá v nalezeném vozidle detekovat RZ. Bylo zapotřebí vyhledat natrénované váhy pro detekci vozidel nebo vytvořit vlastní dataset a trénovat síť na něm. Pro tuto práci byly zvoleny obě varianty.

Pro základní dataset vozidel byl využit Open Images (OI) dataset [31]. OI dataset obsahuje 627 obrázků vozidel pěti kategorií: automobil, autobus, motocykl, nákladní vozidlo a ambulance (obrázek 4.3). Zmíněný dataset nemá všechny kategorie zastoupené rovnoměrně. Označených automobilů má 650 a v ostatních katergoriích okolo 140 značek na kategorii. Nicméně tato nevyváženost by nemusela dělat značný problém, protože práce je soustředěna hlavně na detekci automobilů a jejich registračních značek.

Obrázek 4.3: Ukázka snímků z datasetu Open Images [31].

Modely natrénované na OI datasetu nepřesáhly mAP přes 0,290 (tabulka 4.1). Proto bylo do základního datasetu přidáno dalších 1000 obrázků automobilů z datasetu ze Stanfordovy univerzity a 66 snímků z kamerových záznamů VŠB. Tato úprava zvýšila mAP až na 0,8.

Základní verze datasetu obsahuje celkem 1 814 snímků. Z toho 1 289 v trénovací, 362 ve validační a nakonec 163 v testovací množině.

| Model | mAP |
|---|---|
| YOLOv5s | 0,275 |
| YOLOv5m | **0,290** |
| YOLOv3-tiny | 0,233 |
| YOLOv3 | 0,277 |

Tabulka 4.1: Výsledky detekce vozidel modelů na testovací množině natrénovaných na Open Images datasetu.

### 4.2.3 Augmentované datasety

Protože datasety pro trénování CNN vyžadují velké množství dat, existuje metoda k jejich rozšíření bez nutnosti hledat nové obrázky. Zmíněná metoda se nazývá datová augmentace. Tato metoda je schopna ze stávajících snímků vygenerovat upravené verze aplikováním základních efektů na úpravy obrázků (oříznutí, teplota barev, atd). [26]

Tento postup pomáhá nejen s rozšířením datasetů, ale také s problémem přeučení modelu (model overfitting). Přeučený model je takový, který se naučí trénovací data až moc dobře a má na validačních datech dobré výsledky, ale naopak nemá dobré výsledky na datech mimo dataset na kterém byl trénovaný [32].

Podobně jako ve článku [33] byly i datasety v této práci rozšířeny pomocí datové augmentace. Pro přidání augmentace do základních datasetů RZ a vozidel byla využita webová služba Roboflow [34], která se specializuje na vytváření a editaci datasetů. Možnosti augmentace byly použity pouze na trénovacích množinách (obrázek 4.4).

Níže jsou rozepsány použité efekty augmentace pro datasety v této práci:

- Oříznutí – Přidává variabilitu umístění a velikosti objektů. Napomáhá model naučit se, že hledané objekty nemusí být v obrázku vždy plně viditelné nebo ve stejné velikosti.
- Saturace – Upravuje živost barev. Tato možnost pomáhá modelu detekovat objekty lépe, když se barvy na testovacích obrázcích liší od trénovacích.
- Jas a expozice – Pomáhá modelu být odolnější vůči změnám osvětlení snímků a nastavení fotoaparátu.
- Šum – Člověk může jednoduše ignorovat šum v obrázcích, ale pro modely počítačového vidění je to složitější. Toto je problém tzv adversarial attacks [35], kde malé, člověkem nepostřehnutelné změny pixelů mohou náhle změnit schopnost modelu provádět přesnou predikci.

Na každý vstupní obrázek byl aplikovaný náhodný počet efektů s náhodnou intenzitou. Pro každý obrázek trénovací množiny z RZ datasetu byla vytvořena právě jedna augmentovaná verze, takže velikost trénovací množiny augmentovaného RZ datasetu byla zdvojnásobena na celkem 2 226 obrázků. Pro dataset vozidel byla použita augmentace pouze na část se snímky z kamerových

záznamů VŠB. Těch bylo 66, pro každý z těchto obrázků byly vygenerovány dvě různé verze s aplikovanými efekty. Tento postup přidal 132 nových obrázků do augmentované verze datasetu vozidel, takže trénovací množina tohoto datasetu obsahovala 1 639 obrázků.

Obrázek 4.4: Ukázka snímku z augmentované verze datasetu.

## 4.3 Trénování

Trénování je proces, při němž se model učí rozpoznávat vlastnosti objektů. Každá třída rozpoznávaných objektů má svoje specifické vlastnosti, které napomáhají ke klasifikování objektů dané třídy. Například všechna kola jsou kulatá nebo všechny vnitřní úhly čtverce svírají 90°. [36]

Výstupem trénovaní modelu jsou soubory s váhami. V případě YOLOv3 a YOLOv5 jsou k dispozici po trénovaní dva soubory, jeden s názvem *best.pt* a druhý *last.pt*. Jak již názvy napovídají: last.pt je soubor s váhami, vygenerovaný v poslední trénovací epoše. Dále best.pt obsahuje váhy, které měly při validaci, prováděné po každé epoše, nejlepší výsledek mAP.

Modely byly trénovány jak na osobním počítači s grafickou kartou Nvidia GTX 1660 tak i na službě Google Colaboratory.

### 4.3.1 Google Colaboratory

Google Colaboratory (GC) [37] je služba zdarma od Googlu fungující na podobném principu jako Jupyter Notebook, umožňuje psát a spouštět Python kód přímo v internetovém prohlížeči. V sešitě GC je uživateli po spuštění relace vytvořen nový virtuální počítač s přidělenou grafickou kartou. Uživatelům jsou přidělovány grafické karty z výběru Nvidia K80, T4, P4 a P100. Google v této službě prioritizuje předplatitele, ale také uživatele využívající tuto službu interaktivně nebo pokud v poslední době využívali méně prostředků. Takže se může stát, že pokud uživatel trénuje modely celé dny a nespouští moc příkazů, další den nemusí získat žádné volné GPU.

Život virtuálního počítače v GC je omezený na 12 hodin pro neplatící uživatele a na 24 hodin pro předplatitele. To může být nepříjemné pro časově náročné trénování na hodně epochách, pro-

tože po resetování virtuálního počítače uživatel ztratí všechna data ze sešitu, takže i natrénované váhy modelů. Proto je dobré si s GC sešitem spojit Google Drive a nastavit cestu pro ukládání natrénovaných vah na něj.

I přes výše zmíněné nevýhody je GC skvělý nástroj, který poskytuje značnou pomoc pro trénováni modelů.

### 4.3.2 Nastavení trénování

V této podkapitole bude rozebráno nastavení trénovaní modelů. Trénování probíhalo na RZ datasetech a datasetech s vozidly. Před každým trénováním bylo zapotřebí definovat konfiguraci modelu. Měnilo se různé nastavení trénovaní, jako například počet epoch, rozlišení obrázků, velikosti modelů i množství a zdroje obrázků. Doba trénovaní se lišila v závislosti na zvoleném nastavení a výkonu hardwaru na kterém bylo trénovaní spuštěno. Dohromady bylo natrénováno přes 70 souborů s váhami, z toho 40 bylo pro detekci RZ.

Převážná většina vah byla trénována na GC, protože některé grafické karty této služby dosahují i 2x vyššího výkonu než GTX 1660. Další z důvodů, proč trénování tolik neprobíhalo na osobním počítači je, že někdy po spuštění trénovaní byl tento počítač neovladatelný nebo se trénovaní ani nespustilo. Například při použití argumentu *cache-images*. Tento argument povoluje funkci nahraní použitého datasetu do operační paměti pro zrychlení trénovaní.

Níže je rozepsáno použité nastavení trénování:

#### 4.3.2.1 Velikosti modelů

Z výběru YOLOv5 modelů byly trénovány malé (small), střední (medium) a velké (large). Pro orientaci autoři v názvech modelů použili první písmena z těchto anglických slov. Pro YOLOv3 byly trénovány nejmenší (tiny) a střední velikosti (normal) modelů.

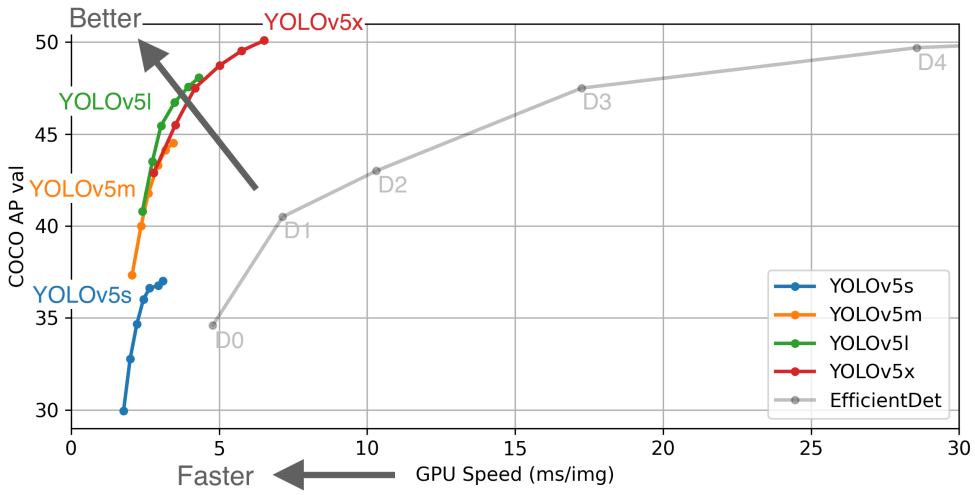
Platí, že větší modely dosahují vyššího AP, ale za cenu pomalejší detekce. Toto chování se dá pozorovat na obrázku 4.5.

#### 4.3.2.2 Rozlišení snímků

Modely byly trénovány na všech čtyřech vytvořených datasetech pro tuto práci. Pro každý dataset se modely trénovaly na standardním rozlišení obrázků 640 px. Dále bylo zvoleno větší rozlišení 960 px a také menší rozlišení 416 px.

#### 4.3.2.3 Batch-size

Protože většinou se nemůže do neuronové sítě na zpracování poslat celý dataset najednou, argument *batch-size* se postará o rozdělení datasetu do částí, které budou postupně zpracovávány [39]. Batch-size se mění i v závislosti na velikosti modelu.




Obrázek 4.5: Porovnání modelů YOLOv5 podle AP a rychlosti detekce (zdroj [38]).

*Batch* = 64 se nejčastěji používá u nejmenších modelů YOLOv5s a YOLOv3-tiny, *batch* = 16 pro střední model YOLOv5m a *batch* = 8 pro největší modely YOLOv5l a YOLOv3-normal.

#### 4.3.2.4 Epochy

Epocha proběhne, když neuronová síť zpracuje jedenkrát celý dataset [39]. Původně byly modely trénovány na 2000 epochách, ale to se časem ukázalo jako zbytečné. Poté byl počet epoch snížen na 1000. Nakonec byla většina modelů trénována na 600 epochách.

# Kapitola 5

# Experimenty

Následující kapitola bude věnovaná experimentům. Pro vyhodnocení a porovnání modelů bylo zapotřebí vytvořit testovací množinu obrázků, ve kterých byly označené oblasti, kde se podle člověka nachází RZ. Testovací množina obrázků se skládá z obrázků z datasetů, které nebyly obsaženy v trénovací množině a ze snímků z kamerových záznamů v areálu VŠB, které rovněž nebyly použity pro trénování modelů. Testovací množina se skládá z celkem 120 obrázků, z toho 80 je z kamerových záznamů VŠB, 20 z datasetu [28] a 20 z datasetu ze Stanfordovy univerzity [30].

Obrázky k testování byly vybírány tak, aby testovací množina byla co nejrozmanitější a daly se na ní pozorovat změny v trénovacích datech, nastavení trénovaní a výběr modelu. Výsledky experimentů budou obsahovat tabulky a komentář.

## 5.1 Detekce RZ

Jako první budou vyhodnoceny experimenty detekce samotné RZ na YOLOv3 a YOLOv5 modelech. Pro změny nastavení trénování nebo datasetu bude většinou zvlášť podkapitola a tabulka. Všechny experimenty byly realizovány na již zmíněné vytvořené testovací množině se 120 obrázky. Modely byly trénovány na 600 epochách a byly použity váhy best.pt. Sloupec T v tabulkách obsahuje průměrný čas experimentu provedený na grafické kartě Nvidia GTX 1660.

### 5.1.1 Základní dataset

Tabulka 5.1 obsahuje výsledky detekce modelů YOLOv3 a YOLOv5 natrénované na základním datasetu, který byl popsán v kapitole o datasetech, s nastavením rozlišení trénovací množiny na 640 px. Detekce byla provedena na rozlišení testovací množiny 640 px a 960 px:

| | Rozlišní snímků | AP | T (s) |
|---|---|---|---|
| YOLOv5s | 640 px | **0,588** | 0,037 |
| YOLOv5m | 640 px | 0,529 | 0,076 |
| YOLOv5l | 640 px | 0,532 | 0,133 |
| YOLOv3-tiny | 640 px | 0,587 | **0,020** |
| YOLOv3-normal | 640 px | **0,588** | 0,153 |
| YOLOv5s | 960 px | 0,709 | 0,049 |
| YOLOv5m | 960 px | **0,739** | 0,115 |
| YOLOv5l | 960 px | 0,709 | 0,215 |
| YOLOv3-tiny | 960 px | 0,701 | **0,030** |
| YOLOv3-normal | 960 px | 0,695 | 0,267 |

Tabulka 5.1: Výsledky detekce RZ modelů trénovaných na základním datasetu s rozlišením 640 px.

#### Analýza výsledků z tabulky 5.1

Podle tabulky 5.1 model YOLOv5m na obrázcích s rozlišením 960 pixelů dosáhl nejlepšího výsledku s $AP = 0,739$. Za ním se o druhé místo dělí modely YOLOv5s a YOLOv5l s $AP = 0,709$, také s detekcí na rozlišení 960 pixelů.

Dále se dá z tabulky 5.1 vyčíst, že rozlišení testovacích obrázků v některých případech značně ovlivňuje rychlost detekce. Například YOLOv3-normal na rozlišení 960 px i 640 px mělo ze všech modelů nejhorší časy $T = 0,267s$ a $T = 0,153s$.

YOLOv5s a YOLOv3-tiny jsou podobně velké sítě a na výsledcích se dá pozorovat, že si jsou svojí přesností detekce hodně podobné. Jejich AP se na stejném rozlišení obrázků liší jen v řádech jednotek. Na druhou stranu YOLOv3-tiny je někdy skoro o polovinu rychlejší než YOLOv5s.

Z tohoto experimentu se dá vyvodit, že vyšší rozlišení obrázků testovací množiny má pozitivní vliv na úspěšnost detekce za cenu vyššího času zpracovávání obrázků.

#### Subjektivní hodnocení

Na konec experimentu je přidáno i subjektivní hodnocení výsledných obrázků:

1. Model YOLOv5m na rozlišení 640 px jako jediný často detekoval otvor pro bankovky na pokladně u závory 1 jako RZ a model YOLOv5l na stejném rozlišení a stejné pokladně detekoval nápis „pokladna“ jako RZ s poměrně vysokou konfidencí 78-83 % (obrázek 5.1a).

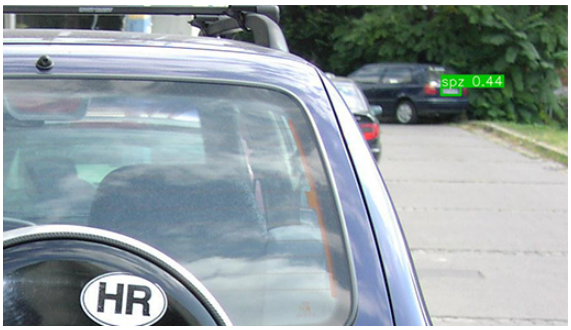
2. Modelu YOLOv5s na rozlišení 960 px se podařilo najít v dálce malou RZ, u které nebylo předpokládáno, že by byla někdy nalezena, takže ani nebyla člověkem označena. Proto tento obdivuhodný výkon modelu byl spíš v jeho neprospěch, protože tato detekce byla označena jako falešně pozitivní (obrázek 5.1b).

3. Model YOLOv5m na rozlišení 960 px našel RZ, která byla celá pokryta sněhem a také bohužel započítaná jako falešně pozitivní (obrázek 5.2a), ze stejných důvodů jako v minulém bodě.

4. Modely YOLOv5l a YOLOv3-tiny na rozlišení 960 px jako jediné detekovaly na závoře 1 cedulku s jejím označením jako RZ (obrázek 5.2b).

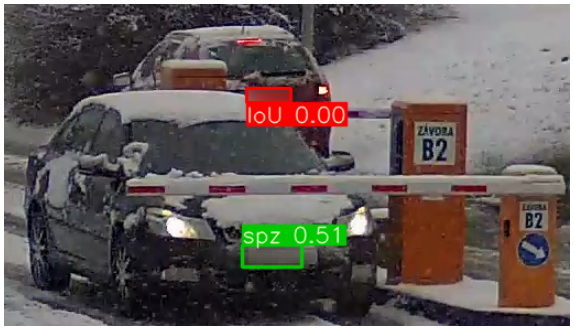
5. Model YOLOv3-normal na rozlišení 960 px jako jediný u závory 4 detekoval svislou dopravní značku „Konec zóny s dopravním omezením“ jako RZ (obrázek 5.3).


(a) Chybná detekce otvoru pro bankovky a nápisu „pokladna“ jako RZ.

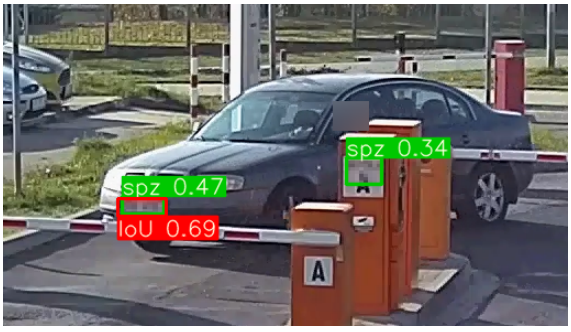



(b) Korektní detekce RZ v dálce.

Obrázek 5.1: Ukázky detekcí.




(a) Korektní detekce zasněžené RZ.




(b) Chybná detekce označení závory jako RZ.

Obrázek 5.2: Ukázky detekcí.


Obrázek 5.3: Dopravní značka „Konec zóny s dopravním omezením“ chybně detekována jako RZ.

### 5.1.2 Změna rozlišení trénovací množiny

Modely v minulém experimentu byly trénovány na obrázcích o velikosti 640 px. Následující dva experimenty ukážou jaký dopad má změna rozlišení obrázku v trenovacích množinách na úspěšnost detekce. Trénování opět probíhalo na základním datasetu. Tabulka 5.2 obsahuje výsledky detekce modelů trénovaných na rozlišení 416 px a 960 px.

V tabulkách budou vynechány řádky experimentů s rozlišením testovací množiny 640 px, protože jak již je známo z minulé podkapitoly, detekce na vyšším rozlišení 960 px dosahují vyššího AP, ale za cenu delšího času detekce. Proto pro lepší přehlednost tabulek není potřeba se detekcí na 640 px, v kapitolách o detekcích samotné RZ, dále zabývat.

| | FP | AP | T (s) |
|---|---|---|---|
| YOLOv5s_416px | 15 | 0,770 | 0,048 |
| YOLOv5m_416px | 10 | **0,802** | 0,115 |
| YOLOv5l_416px | 77 | 0,593 | 0,215 |
| YOLOv3-tiny_416px | 18 | 0,726 | **0,030** |
| YOLOv3-normal_416px | 8 | **0,802** | 0,267 |
| YOLOv5s_640px | 3 | 0,709 | 0,049 |
| YOLOv5m_640px | 3 | 0,739 | 0,115 |
| YOLOv5l_640px | 5 | 0,709 | 0,215 |
| YOLOv3-tiny_640px | 9 | 0,701 | **0,030** |
| YOLOv3-normal_640px | 4 | 0,695 | 0,267 |
| YOLOv5s_960px | 3 | 0,607 | 0,048 |
| YOLOv5m_960px | 5 | 0,643 | 0,115 |
| YOLOv5l_960px | 3 | 0,638 | 0,215 |
| YOLOv3-tiny_960px | 2 | 0,609 | **0,030** |
| YOLOv3-normal_960px | 5 | 0,643 | 0,267 |

Tabulka 5.2: Výsledky detekce RZ modelů trénovaných na základním datasetu s rozlišením 416 px, 640 px a 960 px.

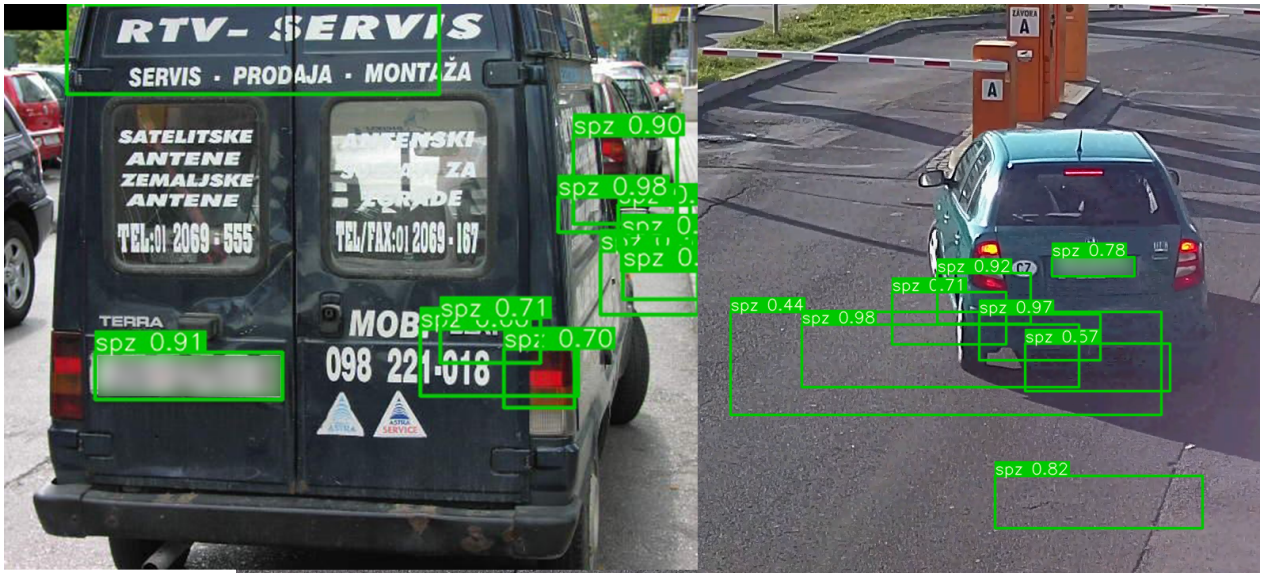
### Analýza výsledků z tabulky 5.2

Překvapivě nejlepších výsledků dosáhly modely trénované na nejnižším rolišení 416 px. Všechny modely trénované na rozlišení 416 px měly nejlepší AP oproti ostatním ve své kategorii. Naopak modely trénované na nejvyšším rozlišení 960 px měly, až na výjimku YOLOv5l_416px, nejhorší AP. Zmíněný model YOLOv5l_416px měl celkem 77 falešně pozitivních detekcí, což je nejvíce ze všech experimentů, které byly provedeny na detekci samotné RZ.

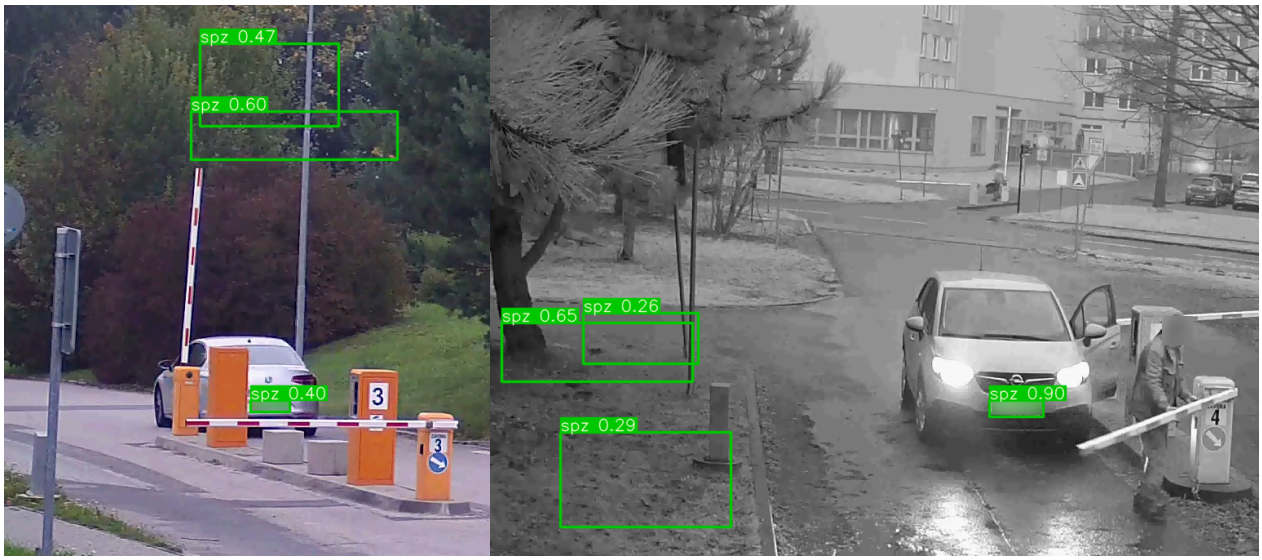
Dále se dá z tabulek vyčíst, že průměrný čas detekce T se u stejně velkých modelů natrénovaných na odlišných rozlišení obrázků neměnil.

### Subjektivní hodnocení

Opět se objevovaly situace, kdy sítě predikovaly jako RZ i jiné objekty. Například světlomety automobilů, značky na závorách atp. Nejzajímavější bylo pozorovat výsledné obrázky detekce již zmíněného modelu YOLOv5l_416px. Ten viděl shluky registračních značek i na čistém asfaltu nebo v korunách stromů.




Obrázek 5.4: Ukázky chybných detekcí RZ uvnitř vozidel a mimo nich.




Obrázek 5.5: Ukázky chybných detekcí RZ v korunách stromů a na travnatých plochách.

### 5.1.3 Různý počet trénovacích epoch

Další experiment se zabývá otázkou, kolik je potřeba trénovacích epoch pro dosažení optimálních výsledků detekce. Tentokrát trénování probíhalo pouze na nejmenších modelech YOLOv3-tiny a YOLOv5s na 200, 400, 600 a 2000 epochách.

| | AP | T (s) |
|---|---|---|
| YOLOv5s_200ep | 0,732 | 0,048 |
| YOLOv5s_400ep | **0,734** | 0,049 |
| YOLOv5s_600ep | 0,686 | 0,048 |
| YOLOv5s_2000ep | 0,709 | 0,049 |
| YOLOv3-tiny_200ep | 0,606 | 0,030 |
| YOLOv3-tiny_400ep | 0,661 | 0,030 |
| YOLOv3-tiny_600ep | 0,676 | 0,030 |
| YOLOv3-tiny_2000ep | **0,701** | 0,030 |

Tabulka 5.3: Výsledky detekce RZ modelů natrénovaných na základním datasetu s různým počtem epoch.

#### Analýza výsledků z tabulky 5.3

Z tabulky 5.3 je zřejmé, že velký počet trénovacích epoch nutně nevede k lepším výsledkům detekce.

YOLOv5 modely měly velice podobné výsledky. V rámci YOLOv5 modelů nejlépe dopadl YOLOv5_400ep s $AP = 0.734$ a nejhůře model YOLOv5s_600ep s $AP = 0.686$.

Naopak u YOLOv3 modelů v tomto experimentu platí, že čím více trénovacími epochami model prošel, tím lepšího AP dosáhl.

### 5.1.4 Upravený dataset

Zatím se popsané experimenty zabývaly jen nastavením trénování, proto se tato kapitola zaměří i na obrázky v trénovacích množinách. Bude porovnán základní a augmentovaný dataset na modelech YOLOv3 a YOLOv5. Augmentovaný dataset má ke každému obrázku v trénovací množině právě jednu upravenou verzi. V kapitole o datasetech je podrobněji rozebrán.

Pro jednodušší orientaci v textu je použita v názvech modelů zkratka *aug* jako augmentovaný.

| | AP | T (s) |
|---|---|---|
| YOLOv5s | 0,709 | 0,049 |
| YOLOv5m | **0,739** | 0,115 |
| YOLOv5l | 0,709 | 0,215 |
| YOLOv3-tiny | 0,701 | **0,030** |
| YOLOv3-normal | 0,695 | 0,267 |
| YOLOv5s_aug | 0,698 | 0,048 |
| YOLOv5m_aug | 0,753 | 0,115 |
| YOLOv5l_aug | 0,731 | 0,215 |
| YOLOv3-tiny_aug | 0,666 | **0,030** |
| YOLOv3-normal_aug | **0,787** | 0,267 |

Tabulka 5.4: Výsledky detekce RZ modelů natrénovaných na základním a augmentovaném datasetu.

#### Analýza výsledků z tabulky 5.4

Podle výsledků úprava datasetu v případě detekce RZ nevedla k jejímu významnému zlepšení ani zhoršení. Augmentovaný dataset zlepšil výsledek detekce pro model YOLOv3-normal, kdy na základním datasetu měl tento model nejhorší AP z YOLOv3 modelů $AP = 0.695$ a stejný model, natrénovaný na upraveném datasetu dosáhl nejlepšího skóre z celé tabulky $AP = 0.787$.

V tomto experimentu se průměrný čas detekce T u stejně velkých modelů nemění.

#### Subjektivní hodnocení

Ani výsledné snímky nevypadají, že by jim úprava datasetu nějak výrazně pomohla. Pouze na pár ojedinělých případech se dalo pozorovat zlepšení. Model YOLOv5s_aug jako jediný detekoval všechny tři RZ na snímku zachyceném za snížené viditelnosti (obrázek 5.6). Dále některé modely trénované na augmentovaném datasetu, dokázaly detekovat RZ překrytou dopravní značkou (obrázek 5.7).


Obrázek 5.6: Ukázka úspěšné detekce RZ na snímku se sníženou viditelností.


Obrázek 5.7: Ukázka úspěšné detekce RZ na snímku, kde svislé dopravní značení překrývá RZ.

### 5.1.5 Předtrénovaný model

V následujícím experimentu se otestují modely, které byly trénovány na předtrénovaných váhách COCO (Common Objects in Context) datasetu. COCO [40] je rozsáhlý dataset obsahující okolo 330 tisíc obrázků, z toho přes 200 tisíc anotovaných a v nich 80 kategorií objektů od automobilu po zubní kartáček.

Trénovaní na předtrénovaném modelu může pomoci neuronové síti se rychleji učit, pokud předtrénovaný model obsahuje stejné třídy objektů jako má dataset na kterém bude trénována. COCO dataset neobsahuje třídu registračních značek, ale i přesto tato možnost byla vyzkoušena alespoň na nejmenších YOLOv3-tiny a YOLOv5s modelech.

Pro srovnání byly do tabulky 5.5 přidány pro obě verze YOLO architektury dva modely stejné velikosti s dosud nejlepším AP z minulých experimentů.

Pro předtrénovaný model bude v názvu použita zkratka *pre*.

| | AP | T (s) |
|---|---|---|
| YOLOv5s | 0,709 | 0,049 |
| YOLOv5s_416px | 0,770 | 0,048 |
| YOLOv5s_pre | **0,785** | 0,048 |
| YOLOv3-tiny | 0,701 | 0,030 |
| YOLOv3-tiny_416px | **0,726** | 0,030 |
| YOLOv3-tiny_pre | 0,669 | 0,030 |

Tabulka 5.5: Výsledky detekce RZ sítí trénovaných na předtrénovaných vahách.

#### Analýza výsledků z tabulky 5.5

I když COCO dataset neobsahuje třídu s registračními značkami, tak model YOLOv5_pre vycházející z jeho předtrénovaných vah měl nejlepší výsledek $AP = 0,785$. Naopak model YOLOv3_pre dosáhl nejhoršího výsledku z celé tabulky ($AP = 0,669$).

### 5.1.6 Shrnutí nejlepších výsledků

V této podkapitole budou k oběma modelům (YOLOv3 a YOLOv5) vypsány 3 výsledky experimentů s nejvyšším AP napříč všemi předchozími podkapitolami.

| | AP | T (s) |
|---|---|---|
| YOLOv5m_416px | **0,802** | 0,115 |
| YOLOv5s_pre | 0,785 | **0,048** |
| YOLOv5s_416px | 0,770 | **0,048** |
| YOLOv3-normal_416px | **0,802** | 0,267 |
| YOLOv3-normal_aug | 0,787 | 0,267 |
| YOLOv3-tiny_416px | 0,726 | **0,030** |

Tabulka 5.6: Experimenty s nejvyšším AP na rozlišení testovací množiny 960 px.

Nejlepšího $AP = 0,802$ dosáhly modely YOLOv5m_416px a YOLOv3-normal_416px. Liší se akorát v čase detekce T, kdy YOLOv5m, protože je menší model, má $2,3\times$ větší rychlost detekce.

## 5.2 Detekce vozdila

Tato kapitola se bude věnovat experimentům detekce samotných vozidel. Pro rozšíření YOLO implementace, kdy nejdříve bude detekováno vozidlo a následně v něm RZ, bude zapotřebí mít dva soubory s váhami. Jeden, který bude naučený detekovat vozidla a druhý, který bude umět detekovat RZ. Korektní detekce vozidla je důležitá z důvodu, že RZ bude možné hledat jen v detekovaných vozidlech.

Na detekci vozidel byl proveden podobný počet experimentů jako na detekci RZ. Avšak z důvodů rozsahu práce pro tuto kapitolu bylo vybráno a popsáno pouze několik nejlepších natrénovaných modelů.

| | Rozlišení snímků | TP | T (s) |
|---|---|---|---|
| YOLOv5s_COCO | 640 px | **203** | **0,038** |
| YOLOv5m_COCO | 640 px | 201 | 0,078 |
| YOLOv5l_COCO | 640 px | 193 | 0,135 |
| YOLOv5l_aug | 960 px | 167 | 0,217 |
| YOLOv5m | 960 px | 160 | 0,117 |
| YOLOv5s | 960 px | 158 | 0,048 |

Tabulka 5.7: Výsledky nejlepších detekcí vozidel.

#### Analýza výsledků z tabulky 5.7

V tabulce 5.7 se vyskytuje sloupec TP udávající počet pravdivě pozitivních detekcí. Sloupec s AP byl vynechán, z důvodu chyby lidského faktoru. Stažené modely natrénované na COCO datasetu detekovaly i mnoho zaparkovaných nebo jen částečně viditelných vozidel, které nebyly označeny člověkem, takže se tyto detekce započítávaly jako falešně pozitivní a výsledné AP z toho důvodu již nebylo relevantní.

Tabulka je seřazena sestupně od největšího TP. Váhy natrénované na COCO datasetu našly nejvíce vozidel. Do nejlepších výsledků se nedostal žádný z YOLOv3 modelů. Modely natrénované na COCO datasetu měly oproti ostatním nejlepší výsledky při rozlišení testovací množiny 640 px.

## 5.3 Detekce vozdila a RZ

Následující kapitola bude věnována experimentům, ve kterých nejdříve jedna neuronová síť detekuje vozidlo a druhá detekuje jeho RZ. Tento experiment si dává za cíl zlepšit detekci malých RZ a odstranění problému falešně pozitivní detekce RZ v oblastech kde se nenachází vozidla.

Pro rozpoznání RZ byly použity všechny váhy modelů z tabulky 5.6 a pro rozpoznání vozidel všechny váhy modelů z tabulky 5.7.

Z důvodu omezené šířky tabulek jsou v některých nadpisech sloupců v tabulce 5.8 použity zkratky, které nyní budou vysvětleny:

- TM1 – průměrný čas detekce na jeden snímek pro model detekující vozidla
- TM2 – průměrný čas detekce na jeden snímek pro model detekující RZ
- TM – průměrný čas detekce na jeden snímek pro oba modely dohromady
- T – celkový čas detekce na testovací množině (celkem 120 snímků)

| Model pro vozidla | Model pro RZ | AP | TM1 (s) | TM2 (s) | TM | T (s) |
|---|---|---|---|---|---|---|
| YOLOv5l_aug | YOLOv3_aug | **0.720** | 0.135 | 0.247 | 0.382 | 56.857 |
| YOLOv5l_aug | YOLOv5s_pre | 0.715 | 0.136 | **0.047** | 0.183 | 33.396 |
| YOLOv5s | YOLOv5s_416px | 0.713 | **0.035** | 0.051 | **0.086** | **22.352** |
| YOLOv5s | YOLOv3_aug | 0.712 | **0.035** | 0.269 | 0.305 | 48.354 |
| YOLOv5m | YOLOv3_aug | 0.711 | 0.078 | 0.247 | 0.325 | 50.695 |
| YOLOv5m_COCO | YOLOv3_aug | **0.735** | 0.079 | 0.767 | 0.846 | 113.827 |
| YOLOv5l_COCO | YOLOv3_aug | 0.710 | 0.143 | 0.764 | 0.907 | 120.992 |
| YOLOv5s_COCO | YOLOv3_aug | 0.696 | **0.036** | 0.640 | 0.676 | 93.367 |
| YOLOv5l_COCO | YOLOv5s_pre | 0.678 | 0.143 | 0.162 | 0.305 | 48.784 |
| YOLOv5m_COCO | YOLOv3-tiny_416px | 0.672 | 0.080 | **0.090** | **0.169** | **32.534** |

Tabulka 5.8: Výsledky nejlepších detekcí vozidel a RZ na rozlišení 640 px.

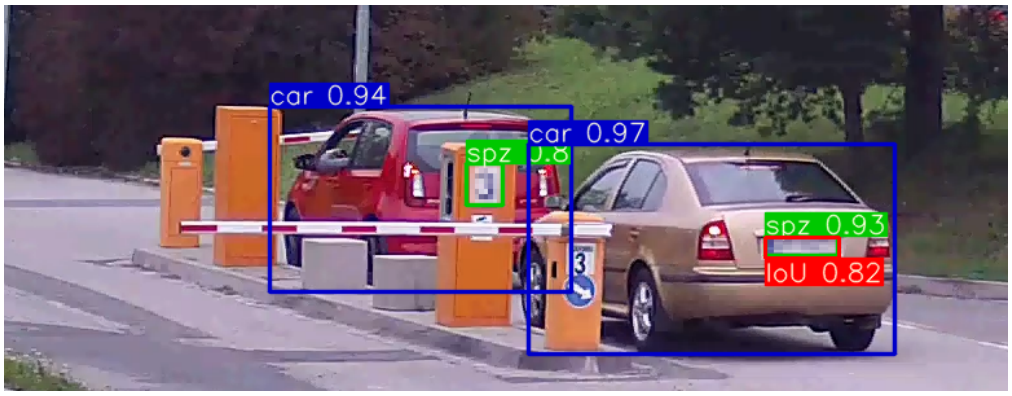
### Analýza výsledků z tabulky 5.8

Oproti experimentům v minulých kapitolách, v této značně dominují detekce na snímcích s rozlišením 640 px. Žádná detekce na rozlišení 960 px se nedostala do tabulek nejlepších výsledků.

Dále se dá z tabulky vyčíst, že celkový čas u všech detekcí modelů trénovaných na COCO datasetu byl delší než časy ostatních modelů. Důvodem je vzrůstající časová náročnost algoritmu s každým nalezeným vozidlem. Protože jak je již známo z předchozí kapitoly (tabulka 5.7), váhy natrénované na COCO datasetu měly nejvíce pravdivě pozitivních detekcí vozidel.

### Subjektivní hodnocení

V těchto experimentech se již tak často nevyskytovaly detekce RZ v místech, kde by neměly vůbec být. Jako jsou například koruny stromů, okna budov nebo prázdná silnice. Avšak stále se vyskytovaly případy, kdy neuronové sítě detekovaly jako RZ objekt, který překrýval detekované vozidlo. Například na obrázku 5.8, kdy jako RZ je detekováno označení čísla závory.




Obrázek 5.8: Příklad chybné detekce, kdy jako RZ bylo detekováno označení závory.

# Kapitola 6

# Závěr

Cílem této práce bylo vytvoření aplikace, která bude detekovat registrační značky pomocí obrazů. Tato práce se nejdříve věnuje nastudování metod pro detekci RZ. Po nastudování metod byla jedna z nich vybrána, upravena a otestována. Byla porovnána úspěšnost detekce na YOLOv3 a YOLOv5 detektorech. Zároveň byly provedeny experimenty na detekci samotné RZ a rozšíření o detekci, kdy je nejdříve detekováno vozidlo a následně v něm RZ. Dále by se aplikace v této práci mohla rozšířit o detekci a rozpoznání znaků v RZ.

V experimentech s grafickou kartou Nvidia GTX 1660 byla dvoufázová detekce pomalejší než jednofázová. Nejlepší průměrný čas jednofázové detekce na jednom snímku je 38 ms (∼26 FPS) a pro dvoufázovou detekci je 86 ms (∼12 FPS). Nejvyšší dosažené AP měla opět jednofázová detekce $AP = 0,802$ oproti $AP = 0,735$ pro dvoufázovou.

Z experimentů vyplývá, že rychlost dvoufázové detekce této aplikace je nedostačující pro použití v reálném čase. Příčinou je zvyšující se časová složitost algoritmu detekce s každým nalezeným vozidlem. Po vhodných úpravách by se výsledná aplikace mohla použít například pro automatické parkovací systémy.

# Literatura


1. DRAGHICI, Sorin. A Neural Network Based Artificial Vision System for Licence Plate Recognition. *International Journal of Neural Systems*. 1997, roč. 08, č. 01, s. 113–126. Dostupné z DOI: 10.1142/S0129065797000148.
2. *Automatic Number Plate Recognition (ANPR)*. 2021. Dostupné také z: https://www.police.uk/pu/advice-crime-prevention/automatic-number-plate-recognition-anpr/.
3. MATTERS, J.E.L.F.T.|. *Bus lanes*. Dostupné také z: https://tfl.gov.uk/modes/driving/red-routes/rules-of-red-routes/bus-lanes.
4. WIKIPEDIA CONTRIBUTORS. *Automatic number-plate recognition — Wikipedia, The Free Encyclopedia*. 2021. Dostupné také z: https://en.wikipedia.org/w/index.php?title=Automatic_number-plate_recognition&oldid=1018055473. [Online; accessed 27-April-2021].
5. GIRSHICK, Ross; DONAHUE, Jeff; DARRELL, Trevor; MALIK, Jitendra. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. Dostupné z arXiv: 1311.2524 [cs.CV].
6. GIRSHICK, Ross. *Fast R-CNN*. 2015. Dostupné z arXiv: 1504.08083 [cs.CV].
7. REN, Shaoqing; HE, Kaiming; GIRSHICK, Ross; SUN, Jian. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. Dostupné z arXiv: 1506.01497 [cs.CV].
8. REDMON, Joseph; DIVVALA, Santosh Kumar; GIRSHICK, Ross B.; FARHADI, Ali. You Only Look Once: Unified, Real-Time Object Detection. *CoRR*. 2015, roč. abs/1506.02640. Dostupné z arXiv: 1506.02640.
9. HOSANG, Jan Hendrik; BENENSON, Rodrigo; SCHIELE, Bernt. Learning non-maximum suppression. *CoRR*. 2017, roč. abs/1705.02950. Dostupné z arXiv: 1705.02950.
10. *Anchor Boxes for Object Detection - MATLAB and Simulink*. Dostupné také z: https://www.mathworks.com/help/vision/ug/anchor-boxes-for-object-detection.html.
11. REDMON, Joseph; FARHADI, Ali. *YOLO9000: Better, Faster, Stronger*. 2016. Dostupné z arXiv: 1612.08242 [cs.CV].

12. REDMON, Joseph; FARHADI, Ali. *YOLOv3: An Incremental Improvement*. 2018. Dostupné z arXiv: 1804.02767 [cs.CV].

13. BOCHKOVSKIY, Alexey; WANG, Chien-Yao; LIAO, Hong-Yuan Mark. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. Dostupné z arXiv: 2004.10934 [cs.CV].

14. LAROCA, Rayson; SEVERO, Evair; ZANLORENSI, Luiz A.; OLIVEIRA, Luiz S.; GONÇALVES, Gabriel Resende; SCHWARTZ, William Robson; MENOTTI, David. A Robust Real-Time Automatic License Plate Recognition based on the YOLO Detector. *CoRR*. 2018, roč. abs/1802.09567. Dostupné z arXiv: 1802.09567.

15. LAROCA, Rayson; ZANLORENSI, Luiz A.; GONÇALVES, Gabriel Resende; TODT, Eduardo; SCHWARTZ, William Robson; MENOTTI, David. An Efficient and Layout-Independent Automatic License Plate Recognition System Based on the YOLO detector. *CoRR*. 2019, roč. abs/1909.01754. Dostupné z arXiv: 1909.01754.

16. LAROCA, R.; SEVERO, E.; ZANLORENSI, L. A.; OLIVEIRA, L. S.; GONÇALVES, G. R.; SCHWARTZ, W. R.; MENOTTI, D. A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector. In: *International Joint Conference on Neural Networks (IJCNN)*. 2018-07, s. 1–10. ISSN 2161-4407. Dostupné z DOI: 10.1109/IJCNN.2018.8489629.

17. JAMTSHO, Yonten; RIYAMONGKOL, Panomkhawn; WARANUSAST, Rattapoom. Real-time Bhutanese license plate localization using YOLO. *ICT Express*. 2020, roč. 6, č. 2, s. 121–124. ISSN 2405-9595. Dostupné z DOI: https://doi.org/10.1016/j.icte.2019.11.001.

18. HENDRY; CHEN, Rung-Ching. Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning. *Image and Vision Computing*. 2019, roč. 87, s. 47–56. ISSN 0262-8856. Dostupné z DOI: https://doi.org/10.1016/j.imavis.2019.04.007.

19. HSU, G.; AMBIKAPATHI, A.; CHUNG, S.; SU, C. Robust license plate detection in the wild. In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2017, s. 1–6. Dostupné z DOI: 10.1109/AVSS.2017.8078493.

20. LIN, C. -H.; LI, Y. A License Plate Recognition System for Severe Tilt Angles Using Mask R-CNN. In: *2019 International Conference on Advanced Mechatronic Systems (ICAMechS)*. 2019, s. 229–234. Dostupné z DOI: 10.1109/ICAMechS.2019.8861691.

21. DONG, Meng; HE, Dongliang; LUO, Chong; LIU, Dong; ZENG, Wenjun. A CNN-Based Approach for Automatic License Plate Recognition in the Wild. In: *BMVC*. 2017.

22. ROSEBROCK, A. *Intersection over Union (IoU) for object detection*. 2021-04-16. Dostupné také z: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

23. WIKIPEDIA CONTRIBUTORS. *Evaluation measures (information retrieval) — Wikipedia, The Free Encyclopedia*. 2021. Dostupné také z: https://en.wikipedia.org/w/index.php?title=Evaluation_measures_(information_retrieval)&oldid=1017908177. [Online; accessed 27-April-2021].

24. JOCHER, Glenn et al. *ultralytics/yolov3: v9.5.0 - YOLOv5 v5.0 release compatibility update for YOLOv3*. Zenodo, 2021-04. Verze v9.5.0. Dostupné z DOI: 10.5281/zenodo.4681234.

25. JOCHER, Glenn et al. *ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations*. Zenodo, 2021-04. Verze v5.0. Dostupné z DOI: 10.5281/zenodo.4679653.

26. POKHREL, S. *Collecting Data for Custom Object Detection - Towards Data Science*. 2020-02-26. Dostupné také z: https://towardsdatascience.com/collecting-data-for-custom-object-detection-e7d888c1469b.

27. *Car License Plate Detection*. 2020-06-01. Dostupné také z: https://www.kaggle.com/andrewmvd/car-plate-detection?select=images.

28. KALAFATIĆ, Z. *Projekt "License Plates"*. 2003. Dostupné také z: http://www.zemris.fer.hr/projects/LicensePlates/english/results.shtml.

29. , T. *tzutalin/labelImg*. Dostupné také z: https://github.com/tzutalin/labelImg.

30. KRAUSE, Jonathan; STARK, Michael; DENG, Jia; FEI-FEI, Li. 3D Object Representations for Fine-Grained Categorization. In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia, 2013.

31. *Vehicles-OpenImages Object Detection Dataset*. 2020-06-19. Dostupné také z: https://public.roboflow.com/object-detection/vehicles-openimages.

32. BROWNLEE, J. *How to Avoid Overfitting in Deep Learning Neural Networks*. 2019-08-06. Dostupné také z: https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/.

33. SILVA, Sergio Montazzolli; JUNG, Claudio Rosito. License Plate Detection and Recognition in Unconstrained Scenarios. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018-09.

34. *Roboflow: Everything you need to start building computer vision into your applications*. Dostupné také z: https://roboflow.com/.

35. JAIN, A. *Breaking neural networks with adversarial attacks - Towards Data Science*. 2020-01-06. Dostupné také z: https://towardsdatascience.com/breaking-neural-networks-with-adversarial-attacks-f4290a9a45aa.

36. WIKIPEDIA CONTRIBUTORS. *Object detection — Wikipedia, The Free Encyclopedia*. 2021. Dostupné také z: https://en.wikipedia.org/w/index.php?title=Object_detection&oldid=1010314280. [Online; accessed 27-April-2021].

37. *Colaboratory – Google*. Dostupné také z: https://research.google.com/colaboratory/faq.html.

38. *Release v4.0 - nn.SiLU() activations, Weights and Biases logging, PyTorch Hub integration · ultralytics/yolov5*. Dostupné také z: https://github.com/ultralytics/yolov5/releases/tag/v4.0.

39. SHARMA, S. *Epoch vs Batch Size vs Iterations - Towards Data Science*. 2019-03-05. Dostupné také z: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9.

40. LIN, Tsung-Yi et al. Microsoft COCO: Common Objects in Context. *CoRR*. 2014, roč. abs/1405.0312. Dostupné z arXiv: 1405.0312.

# Příloha A

# Seznam přiložených souborů a složek

Z důvodu omezené velikosti archívu se v něm nachází pouze malý vzorek obrázků z datasetů a také pouze několik natrénovaných nejmenších modelů. Dále v testovací množině jsou z důvodu ochrany osobních údajů odstraněny snímky kamerových záznamů z vjezdů do areálu VŠB, tyto snímky lze na vyžádání zaslat. V přiloženém archívu se nachází následující seznam souborů a složek:

```
/
├── Aplikace ... složka se zdrojovými kódy aplikace
│   ├── models ... složka obsahující konfigurační soubory modelů
│   ├── runs
│   │   ├── detect ... složka pro ukládání výsledků testování
│   │   └── train ... složka obsahující natrénované modely
│   ├── detect.py... soubor pro testování modelů
│   └── my_test.py ... soubor pro automatizované testování modelů
├── test_data ... složka se snímky, na kterých bylo prováděno testování
├── Ukázky datasetů ... vzorek obrázků z datasetů
├── Výsledky experimentů
│   ├── Detekce SPZ ... ukázka výsledných snímků detekce
│   ├── Detekce vozidla a SPZ ... ukázka výsledných snímků detekce
│   └── Hromadné experimenty ... CSV soubory s výsledky experimentů
└── readme.txt ... soubor s vypsanými příkazy pro spuštění testování
```